Using the BBjRecordSet

By Chris Hardekopf

The new ***BBjRecordSet*** object enables developers to attach GUI controls to underlying data fields that exist in both BBj® files and SQL query result sets. These attached fields make modifying records easy and have the added benefit of automatically changing their contents to reflect the current record changes. In short, the BBjRecordSet simplifies the implementation of existing data capabilities. This article provides a preview of the functionality available in the BBjRecordSet object with the upcoming BBj 4.0 release.

The BBjRecordSet makes the developer's job much easier. It automates common data operations that previously required a programmer to write code, and allows quick and concise updates, insertions, and modifications of records using code and databound GUI controls. Additionally, the BBjRecordSet lets developers abstract the data handling code by providing a common interface for both file and SQL access. The SQL access lets developers incorporate data from any JDBC or ODBC capable database into a BBj application. By combining the BBjRecordSet and ***BBjNavigator*** control, the developer only needs to write data validation routines.

The first step to using the BBjRecordSet is determining what data to use - i.e. which file or what SQL query is appropriate. The second step is to create a BBjRecordSet object using a method of the ***BBjAPI***. At this point, the developer should create and display the window, then bind the fields from the BBjRecordSet to the associated GUI controls. The final step entails placing a BBjNavigator control on the window and registering callbacks for the navigator events. The callbacks should check to see if the BBjRecordSet is dirty (the underlying data is modified and no longer matches the BBjRecordSet), perform any desired validation to ensure data consistency, and save changes to the record before advancing to the next record.

Obtaining the current ***BBjRecordData*** from the BBjRecordSet provides access to a copy of the current record. The developer can then get and set the various field values using the BBjRecordData. After modifying the record, the developer can determine whether to save the updated version to the disk, overwriting the original record. If so, the developer uses this BBjRecordData by passing it as an argument to the update method of the BBjRecordSet.

When using a BBjRecordSet in combination with BBj files, a record on disk might change before the developer attempts an update through the BBjRecordSet. In this case, the BBjRecordSet update method returns an error. In response to this error, the developer can choose one of a number of possible actions. The first possibility is to discard the attempted changes. The second possibility is to retrieve the current contents of the record from the source, selectively decide which changes to make, and then once again attempt the update. Usually the developer will decide to implement some variation on the second approach. Then, depending on the application, the developer determines how to select and merge any changes and how automated to make this process.

The BBjRecordSet also has the ability to seek forward and backwards for records based on field contents. Seeking is useful for locating particular records inside of the larger set represented by the BBjRecordSet as a whole. In order to use this functionality, the developer must simply place the desired field values into a BBjRecordData object and tell the BBjRecordSet to seek either ***forward*** or ***backward*** for the data. This enables the developer to find all instances of the specified field values by starting at the beginning and repeatedly seeking forward until reaching the end of the BBjRecordSet. Depending on the size of the BBjRecordSet and its field values, seeking could take some time since it must search through every record in the BBjRecordSet.

The ***sample source code*** illustrates the structure of a BBjRecordSet program. An important part for most developers is the validation of record data before changes are saved to the file or database. As shown by the **`"VALIDATE_SAVE"`** label, the developer can view and possibly change the field values of the record before the saving the changes. After processing the record, the developer can then choose whether the program should move to the next record.

Developers can also use the BBjRecordSet in a more automated fashion. See the ***BBj Databound Controls*** for information about how to use ***ResBuilder***® and ***GUIBuilder***® to create applications that use BBjRecordSet in a way that is more automatic and GUI oriented.